Fundamenta Informaticae 72 (2006) 1–28
IOS Press

Nondeterministic Bimachines and Rational Relations with Finite Codomain

Nicolae Santean* **and Sheng Yu**
Computer Science Department
University of Western Ontario
London, ON N6A 5B8, Canada
nic@csd.uwo.ca; syu@csd.uwo.ca

Abstract. Bimachines are important conceptual tools used for the characterization of rational word functions (realized by single-valued transducers). Despite the attention received in the past, these sequential machines are far from being exhaustively studied. A natural question which has not been addressed so far is what family of transductions are realized by bimachines that operate nondeterministically. We show that these machines characterize input-unambiguous (IU) rational transductions, i.e., those transductions that can be written as a composition of rational functions and finite substitutions. Two more families of rational transductions are defined and related in a natural way to IU transductions: input-deterministic transductions and rational transductions with finite codomain (FC). We have shown that FC transductions are recognizable and that they can be expressed as finite union of subsequential functions. Moreover, they can be realized by nondeterministic bimachines. Finally, we have defined the so called restricted nondeterministic bimachines and shown that, surprisingly, they are more powerful than nondeterministic bimachines: they characterize exactly the family of finitely ambiguous rational transductions.

Keywords: Finite transducer, unambiguous automaton, nondeterministic bimachine, rational function, finite substitution

1. Introduction

The interest in finite (or otherwise) transducers dates back to the early days of Automata Theory. The reason why today there does not exist an established "Theory of Transductions" is that transducers can

* Address for correspondence: Computer Science Department, University of Western Ontario, London, ON N6A 5B8, Canada

arguably be viewed as a generalization of the concept of automata – in the classical sense. Despite this view, transducers lack the deserved exposure in literature. In the past, there have been just a handful of attempts to systematically approach the topic, as for example the work of Eilenberg ([10]) or Berstel ([2]). However, since the 60's, many sporadic results on finite and pushdown transducers have been communicated, becoming part of a rich folklore that is waiting to be organized as a stand-alone theory. To mention just a few, we recall the major early work of Schutzenberger (for example, [26], [25], [27], or [28]), the work of McKnight, Elgot and Nivat ([15], [11] and [19]); then, in the late 70's, the work of Choffrut ([5], [6], or [7]), in the 80's the work of Culik, Berstel, and Reutenauert ([9], [3], [20], [21]), or more recently, the work of Mohri ([16], or [17]). Most recent developments in the area can be found in [4], [1], [13] [8], [18] and other traceable sources. A classical viewed on transducers can be grasped from [14] and a methodical study of the topic can be found in [23].

Why transducers? In the classical theory, automata are viewed as acceptors, or devices that define formal languages. In other words, an automaton is a finite specification of a language. However, in most practical matters, the mere acceptance (or rejection) of words that belong (or do not belong) to a language is limitative. Most of the time we are interested in processing these words in different ways, and this is what transducers do. Simplistically, transducers can be viewed as automata that write a "trace" of their computation on an output tape. This trace can be designed, for example, to represent a processing of the input word. We say that the transducer is performing a translation, or a transduction. With respect to the type of the underlying automaton, we have different types of transducers: pushdown, finite, and these can be single-valued, sequential, subsequential, etc. Arguably the most popular transducers are the finite ones (finite automata with output) which enjoy a plethora of nice algebraic properties. However, it is in our perception that the hierarchy of rational transductions/relations (those realized by finite transducers) is still too coarse, and new types of finite transducers are yet to be discovered.

Arguably one of the most intriguing machines that realize rational transductions are the bimachine, designed by Schutzenberger ([26]) and studied, among others, by Eilenberg who stated their importance in [10, §11.7, Theorem 7.1, p. 321]. A bimachine is a compact representation of the composition of a left and a right sequential transducer, and it characterizes the family of rational functions. A few variations of the original design have been studied in [24], where it has been shown that the scanning direction of its two reading heads does not matter. In this paper we study another variation on the same theme, that is, a bimachine that operate nondeterministically. We show that these machines characterize the family of transductions that can be written as a composition of a rational function and a finite substitution. They are equivalent to the so-called input-unambiguous transducers (IU), which are close relatives of the classical unambiguous transducers. We do also show that nondeterministic bimachines can "simulate" (i.e., they give a representation of) rational relations with finite codomain (FC). Surprisingly, we prove that FC transductions belong strictly to the family of recognizable relations and that they can be written as a finite union of subsequential functions. We noticed that nondeterministic bimachines are a compact representation of the composition of a left sequential transducer and a right input-deterministic transducer – which is a close relative of the classical right sequential transducer. Finally, we have defined restricted nondeterministic bimachines, those who do not reset themselves at each computation step. Surprisingly, we observed that this restriction increases their representation power, allowing them to characterize the entire family of finitely ambiguous (FA) rational relations. Basically, the reset/no-reset dichotomy reveals the difference between IU and FA families.

The paper is structured as following. In the first section we recall a few basic notions (as finite transducers and rational/recognizable transductions) and introduce a few notations. In Section 3 we define

input-unambiguous (IU) transducers and study their properties. We place the family of IU transductions among other already known families in Section 4. It is in this section where we also study rational transductions with finite codomain. In Section 5 we define various types of nondeterministic bimachines and we give an important characterization of IU transductions: they are exactly those realized by nondeterministic bimachines. Finally, in Section 6, we define restricted nondeterministic bimachines and we prove that they characterizes the family of finitely ambiguous rational relations. A few comments and further work are underlined in the last section.

2. Preliminaries

In the following we assume known basic notions of automata theory ([12], [29]). By DFA and NFA we understand deterministic and nondeterministic finite automata, and by ϵ-free NFA we understand NFA with no ϵ-transitions, where ϵ denotes the empty word.

The notion of ambiguity for automata relates to the number of possible successful computations performed by an automaton for a given input. For example, a DFA is unambiguous, whereas an NFA can have various degrees of ambiguity, including zero.

Definition 2.1. An ϵ-NFA A is unambiguous (UNFA) if each word is the label of at most one successful computation in A.

Let X be an alphabet, $A = (Q, X, \delta, q_0, F)$ be an ϵ-free UNFA and for any $S \subseteq Q$ denote

$$\delta^{-1}(S, w) := \{q \in Q \mid \delta(q, w) \cap S \neq \emptyset\}.$$

It is easy to check the following property:

$$\forall u, v \in X^* : \ |\delta(q_0, u) \cap \delta^{-1}(F, v)| \leq 1. \tag{1}$$

In the following we may encounter `lazy` automata, i.e., finite automata whose transitions are labelled with words rather than letters or ϵ. This extension is useful, making easier to use properties of finite automata when dealing with transducers.

By a finite transducer over the alphabets X and Y we understand an automaton over the product of free monoids $X^* \times Y^*$. In other words, a transducer is a finite automaton whose transition labels are elements of $X^* \times Y^*$, with the meaning that the first component of the label is an input word and the second component is an output word. Transducers can also be viewed as two-tape automata, where both components of a transition label are viewed as an input pair of words. It is well known that finite transducers realize `rational word relations` (see for example [23, §IV.1.2, p. 566]), denoted by $Rat(X^* \times Y^*)$, or simply Rat when the alphabets are understood.

Formally, a transducer is a tuple $T = (Q, X, Y, E, q_0, F)$, where Q is a set of states, X, Y are alphabets, q_0 is an initial state, F is a set of final states and E is a `finite` set of transitions which are elements in $Q \times X^* \times Y^* \times Q$. The transduction (binary word relation) realized by T will be denoted by $|T| : X^* \to Y^*$ and is defined similarly to the language accepted by an NFA. The transducer T is `normalized` if the following conditions hold:

1. $E \subseteq Q \times (X \cup \{\epsilon\}) \times (Y \cup \{\epsilon\}) \times Q$;

2. $F = \{q_f\}, q_f \neq q_0$;

3. $(p, x, \alpha, q) \in E \Rightarrow p \neq q_f, \;\; q \neq q_0$.

It is known that any rational transduction is realized by a normalized finite transducer and that any transducer can algorithmically be normalized.

Example 2.1. In Figure 1 are depicted two finite transducers which both realize the function

$$\forall n \geq 1 : \;\; \tau(x^n) = \begin{cases} a^n, & if \;\; n \;\; is \;\; even, \\ b^n, & otherwise. \end{cases}$$

The transducer T' is the normalization of T.

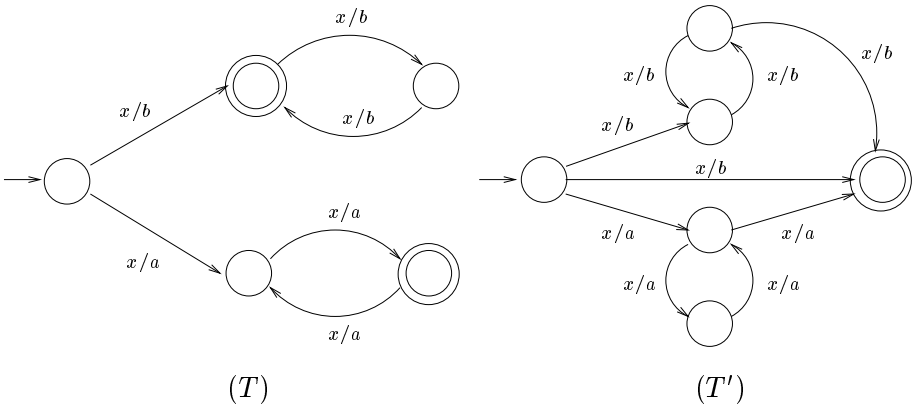

Figure 1. The normalization of transducer T.

By a `useful` state (or transition, path, loop, etc.) in a transducer we understand a state (or transition, path, loop, etc.) which is used in at least one successful computation. By an ϵ-`input loop` we understand a loop (in the transition graph) whose transitions have only ϵ-input labels.

We recall that a recognizable set in a monoid is a set defined by an action over that monoid (see, for example, [23, p. 252]). Recall also that a subset of the direct product of two monoids (also a monoid) is recognizable if and only if it can be written as a finite union of blocks (a block is a direct product of two recognizable sets). This characterization is known as Mezei's characterization of recognizable sets in direct product monoids. In the following, by $Rec(X^* \times Y^*)$ we understand the set of recognizable transductions over the alphabets X and Y, i.e., the family of recognizable subsets of $X^* \times Y^*$.

3. Input-Unambiguous Finite Transducers

It is often useful to see a finite transducer as a pair of automata with the same transition graph, but with different transition labels (in some sense, one may say that these automata are transition-synchronized). If we remove the output labels of a transducer we obtain one automaton (the `input automaton`), and if

we strip the input labels we obtain the other automaton (the `output automaton`). Of our interest is the former, as defined in the following.

Definition 3.1. Let $T = (Q, X, Y, E, q_0, F)$ be a finite transducer. The `input automaton` of T is the finite automaton $A = (Q, X, \delta, q_0, F)$, where δ is given by

$$\forall x \in X^* : \ \ q \in \delta(p, x) \Leftrightarrow \exists \alpha \in Y^* : (p, x, \alpha, q) \in E, \ \text{ where } p, q \in Q.$$

If the transducer is normalized, then its input automaton is an ϵ-NFA, otherwise it is a lazy NFA.

Definition 3.2. A finite transducer T is called `input-unambiguous` (IU, for short) if its input automaton is unambiguous (i.e., an UNFA).

Notice carefully that a transducer can still have different successful paths with a same input labels and nevertheless be input-unambiguous. A such situation is depicted in Figure 2. Notice also the difference

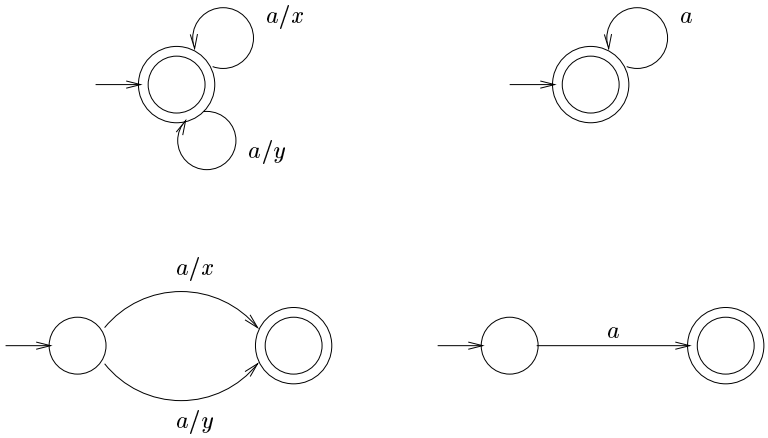

Figure 2. IU transducers (left) and their unambiguous input automata (right).

between this definition and the classical definition of `unambiguous transducers`. Conform [2], an unambiguous transducer is a transducer such that any input word is the input label of at most one successful computation. The following example clarifies this point.

Example 3.1. In Figure 3 we denote by $\alpha, \beta, \gamma, \delta$ arbitrary words in Y^*. The transducer is IU despite the fact that it has more than one successful computation triggered by the input ab. However, it is not unambiguous in the sense of [2, §IV.4, p. 114]. More clearly, an IU transducer is a small variation from the classical definition of unambiguous transducer, that is, we allow the existence of transitions as those labelled b/δ and b/β in Figure 3. At this early stage one can already anticipate that a such transducer realizes the composition of a rational function and a finite substitution. However, we choose to derive this observation as a consequence of the equivalence of IU transducers and nondeterministic bimachines – proven later, in Section 5.

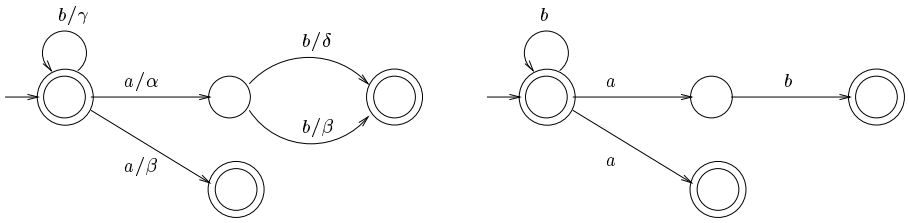

Figure 3. Another IU transducer and its input automaton.

Remark 3.1. An IU transducer cannot have useful ϵ-input loops, in the same way as an unambiguous automaton cannot have useful ϵ-loops. This impossible situation is depicted in Figure 4, where it is clear that the input uv triggers in the input automaton an infinite number of different successful computations.

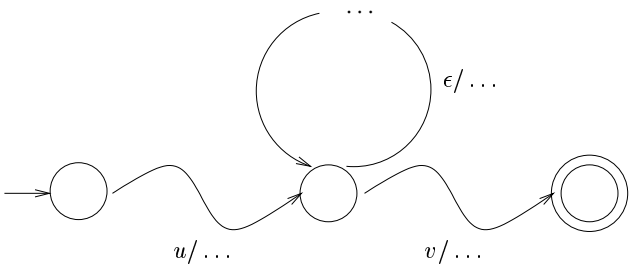

Figure 4. An IU transducer can not have loops as above.

An IU transduction is a transduction realized by an IU transducer. In the following, we give a useful normalization of IU transducers. First notice that given an arbitrary IU transducer, there exists an equivalent IU transducer in normal form, in the sense mentioned at page 3. Indeed, the standard normalization algorithm (see for example [2, §III.6, Corollary 6, p. 79]) does not alter the degree of ambiguity of a transducer.

Recall that a `trim transducer` has only useful states. Without loosing generality, we follow the convention that if the initial state of a transducer is also final then the pair (ϵ, ϵ) is realized by the transducer. This convention has a theoretical explanation which we choose to ignore here, due to its interference with the definition of ambiguity and normalization.

Lemma 3.1. Any IU transduction $\tau : X^* \to Y^*$ with $\tau(\epsilon) = \epsilon$ or $\tau(\epsilon) = \emptyset$ is realized by a trim IU transducer $T = (Q, X, Y, E, q_0, F)$ which satisfy the following conditions:

(i) $E \subset Q \times X \times Y^* \times Q$;

(ii) if $\tau(\epsilon) = \epsilon$, then $F = \{q_0, q_f\}$, else $F = \{q_f\}$, and $q_f \neq q_0$;

(iii) $(p, x, \alpha, q) \in E \Rightarrow q \neq q_f, p \neq q_0$.

Proof:
We first tackle the case where $\tau(\epsilon) = \emptyset$. Let $T = (Q, X, Y, E, q_0, \{q_f\})$ be a trim and normalized (in the sense mentioned at page 3) IU transducer with $|T| = \tau$. In the following we transform T into a transducer obeying the conditions of the lemma by removing the ϵ-input transitions of T without changing the non-ambiguity of the input automaton, as following:

1. `ϵ-input closure`: if (p, ϵ, α, q) and (q, ϵ, β, r) are transitions in E then we add the transition $(p, \epsilon, \alpha\beta, r)$ to E. Since the transducer has no ϵ-input loops (see Remark 3.1), the process eventually stops.

   An important observation at this point (after the closure) is that, since $\tau(\epsilon) = \emptyset$, T cannot have transitions of type $(q_0, \epsilon, \alpha, q_f)$, with α an arbitrary word in Y^*.

2. `final state compensation`: for each pair of transitions (r, a, α, p) and $(p, \epsilon, \beta, q_f)$ we add the transition $(r, a, \alpha\beta, q_f)$. After all pairs have been processed, we remove all transitions $(p, \epsilon, \beta, q_f)$ (recall that q_f has no output transitions). This process allows us to avoid the addition of extra final states, and is depicted in Figure 5.

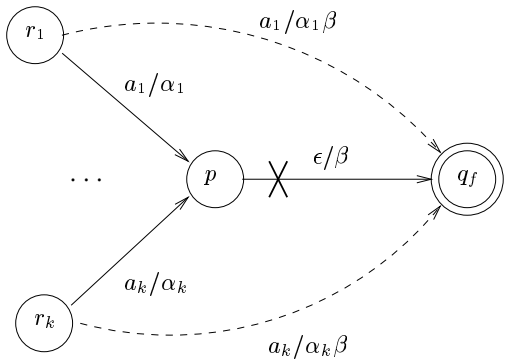

Figure 5. Final state compensation for ϵ-input removal.

3. `ϵ-input transition compensation`: for any combination of transitions (p, ϵ, α, q), (q, a, β, r) with $a \in X$ we add the transition $(p, a, \alpha\beta, r)$ to E.

4. `ϵ-input removal`: we remove all transitions (p, ϵ, α, q) and we perform a trimming.

Notice that the removal at step 4 eliminates any input ambiguity possibly induced in the previous steps. One can easily check that the obtained transducer is trim and IU, and verifies the conditions of the lemma.

The case when $\tau(\epsilon) = \epsilon$ is solved similarly, by discarding all ϵ-input transitions between q_0 and q_f, after step 1. Then, at the end we set the initial state to be also final. □

Remark 3.2. Notice that it is decidable whether a finite transducer is IU or not. The decision can be reduced to whether an ϵ-NFA is UNFA or not. For completeness, we give here a simple proof.

Proof:
Let T be a finite transducer and denote by $A = (Q, \Sigma, \delta, q_0, F)$ the input automaton of T. We define a relation among the states of A, as following. We say that p and q are `siblings`, denoted by $p \sim q$, if and only if there exists a word w such that $p, q \in \delta(q_0, w)$. In other words, p and q are siblings if there exist two computations with the same label and ending in p and q, respectively. Notice that this relation is reflexive, symmetric but not transitive. Notice also that this relation can algorithmically be computed. Indeed, if we perform a standard set construction (as in the NFA determinization) any two states which belong to a same `superconfiguration` are siblings, and only those.

For each state p, denote by R_p the language obtain by setting p to be the initial state in A. Then A is unambiguous if and only if for each two different sibling states p and q, we have $R_p \cap R_q = \emptyset$. Clearly, this condition is algorithmically computable. □

4. A Hierarchy of Ambiguity

In order to put IU transductions into a proper context, in the following we recall two known families of rational transductions: finitely and uniformly ambiguous.

Definition 4.1. A rational transduction $\tau : X^* \to Y^*$ is `finitely ambiguous` (FA) if $|\tau(u)| < \aleph_0, \forall u \in X^*$. We say that τ is `uniformly ambiguous` (UA) if there exist a constant N such that $|\tau(u)| < N, \forall u \in X^*$.

These families of transductions have been studied in the past. For example, it is known that an UA rational transduction can be written as a finite union of rational functions, and it is decidable whether a rational transduction is in FA (this is equivalent to detecting non-trivial ϵ-input loops in a finite transducer). However, we are not aware of whether it is decidable if a rational transduction is in UA or not.

Next we aim at finding the relationship between these families of rational word relations.

Corollary 4.1. $IU \subset FA$.

Proof:
It is a direct consequence of Remark 3.1: since an IU transducer has no ϵ-input loops, any input word can trigger a finite number of words to be written on the output tape. □

Corollary 4.1 affirms that the transductions realized by IU transducers are in FA. However, they are not necessarily in UA. Indeed, the following example shows an IU transducer T which realizes a transduction that is not uniformly ambiguous.

Example 4.1. The transducer in Figure 6 realizes the transduction τ given by:

$$\forall n \geq 1 : \ \tau(a^n) = \begin{cases} \bigcup_{i=1}^{n} \{x^i\}, & \text{if } n \text{ is even,} \\ \bigcup_{i=1}^{n} \{y^i\}, & \text{otherwise,} \end{cases}$$

which clearly is not UA, however it is IU.

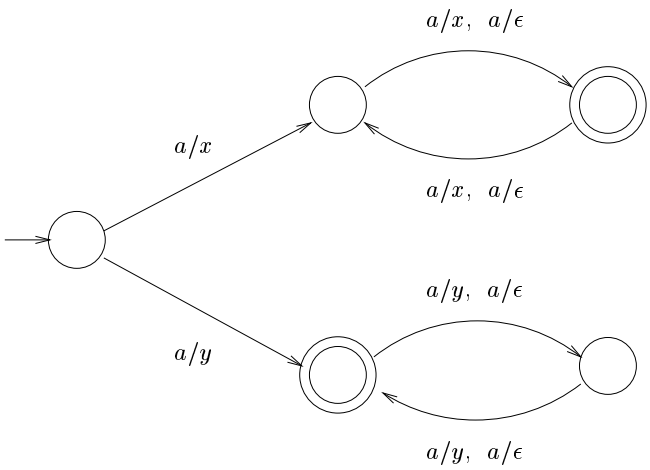

Figure 6. An IU transducer whose transduction is not UA.

On the other hand, not all rational transductions which are UA are necessarily IU. The transduction

$$\tau = \{(a^n, x^n) \mid n \geq 1\} \cup \{(a^n, y^n) \mid n \geq 1\} \tag{2}$$

(with a, x, y different letters) is UA (notice that it is written as a union of two rational functions), however it is not IU. Indeed, a transducer T realizing τ must have two successful computations for each input word a^n: one outputting x^n and the other y^n, for all integers n. If these two successful computation coincide in the input automaton of T, then in T must exist a successful computation which "shuffles" x and y on the output tape, hence T can not be IU.

To the families of transductions discussed so far (FA, UA, and IU) we add a new one, that of rational transductions with finite codomain.

Definition 4.2. A rational transduction $\tau : X^* \to Y^*$ is with finite codomain (FC) if $|\tau(X^*)| < \aleph_0$.

Notice that the condition is equivalent to saying that there exist a constant N such that $\forall v \in \tau(X^*) : \sharp(v) \leq N$, where by $\sharp(v)$ we understand the `length` of the word v, i.e., the number of its letters (accordingly, $\sharp(\epsilon) = 0$).

Obviously, it is decidable whether a rational transduction is in FC or not (it is equivalent to deciding whether the output automaton of a transducer accepts a finite language or not).

Lemma 4.1. A rational transduction $\tau : X^* \to Y^*$ is in FC if and only if it can be written as

$$\tau = \bigcup_{i \in I} [L_i \times R_i],$$

where I is finite, $\{L_i\}_{i \in I}$ are disjoint regular languages, and $\{R_i\}_{i \in I}$ are finite languages.

Proof:
The "if" part of the proof is trivial, since τ is recognizable, hence it is rational, and in addition it has a finite image. We prove the remaining implication.

Let $\tau : X^* \to Y^*$ be a transduction in FC. This implies that $|\tau(X^*)| < \aleph_0$, and consider $\tau(X^*) = \{\alpha_1, \ldots, \alpha_k\} \subset Y^*$, with k integer. Let $T = (Q, X, Y, E, q_0, \{q_f\})$ be a trim, normalized finite transducer realizing τ. Since $\tau \in FC$, the output of any loop in T is the empty word, since otherwise the output image of T would be infinite (in other words, the output automaton of T accepts a finite language, i.e., it can have only ϵ-loops). A view of how the transition graph of T may look like is depicted in Figure 7.

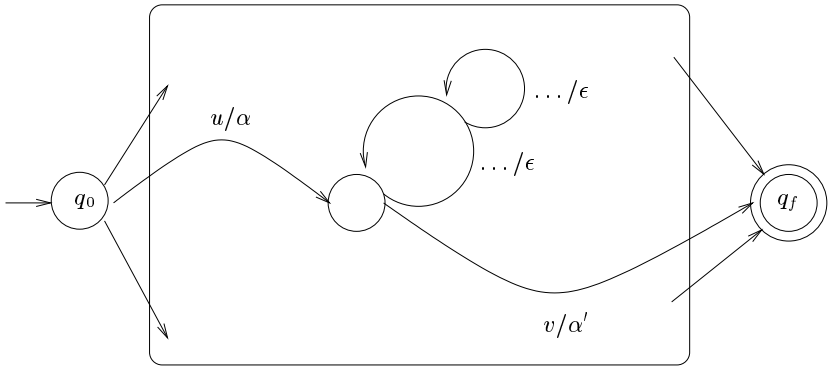

Figure 7. The transition graph of a normalized FC transducer.

By a `simple successful computation` (ssc, for short) in T we understand a computation, from the initial state to the final state, in which no state is repeating. If the initial state of T is also final, we consider $(q_0, \epsilon, \epsilon, q_0)$ to be a ssc as well. It is clear that T (or any other transducer) can perform only a finite number of such computations. Denote by $\{\pi_1, \ldots, \pi_m\}$ the set of ssc in T. Since the loops in T output only the empty word (by a previous observation) and since T is normalized, it is clear that $\pi_1, \ldots, \pi_m$ output, together, exactly $\{\alpha_1, \ldots, \alpha_k\}$, i.e., the entire image of τ.

In the following, we `segregate` $\pi_1, \ldots, \pi_m$ from each other (by viewing them as paths in the transition graph) and attach to each of them copies of the loops which they initially had. The process has an awkward formalization, therefore we choose to rather illustrate it in Figure 8.

In Figure 8 (A), we have a transducer with two simple successful paths: one with input-label uvv' and the other with input label uvv''. The transducer outputs the words $\alpha\beta\gamma$ and $\alpha\beta\gamma'$, which are exactly the outputs of the simple successful computations. We proceed by segregation and we obtain two transducers, each with only one ssc. They are depicted in Figure 8 (B) and (C), and they were obtained by considering each simple successful computation alone, and attaching to it all the loops which it initially had attached.

After the segregation we obtain m transducers $T_1, \ldots, T_m$, each corresponding to a ssc in T. It is clear that

$$| T | = \bigcup_{i=1}^{m} | T_i |,$$

and that each transducer T_i outputs a single word $w_i \in \{\alpha_1, \ldots, \alpha_k\}$. Clearly, $m \geq k$. Denote by $A_1, \ldots A_m$ the input automata of $T_1, \ldots, T_m$, and by $L_1, \ldots, L_m$ their accepted languages. We define

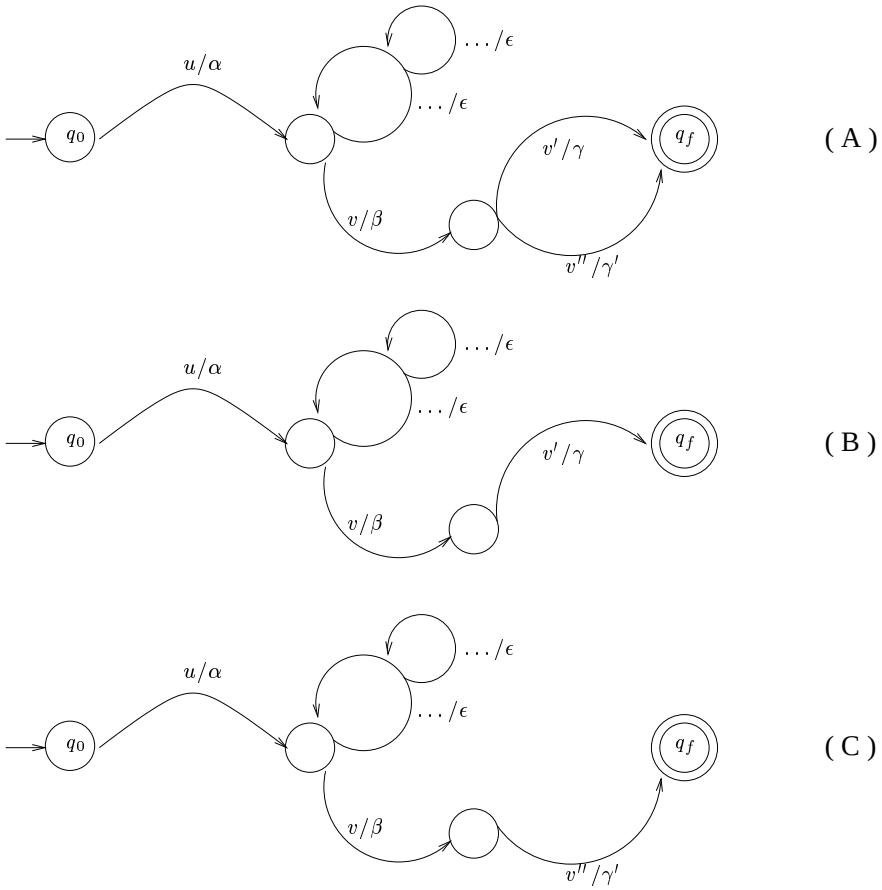

Figure 8. The segregation of simple successful paths.

an equivalence over $\cup_{i=1}^{m} L_i$ as follows:

$$u \equiv v \;\Leftrightarrow\; \forall i, j \in \{1, \ldots, m\} : \; u \in L_i, v \in L_j \Rightarrow \{u, v\} \subset L_i \cap L_j.$$

It is easy to see that $\equiv$ is an equivalence of finite index, and each of its equivalence classes is an intersection of some regular languages in $\{L_1, \ldots, L_m\}$. For example, in Figure 9 we depict a situation where $m = 3$ and $c_1, \ldots, c_7$ are the equivalence classes of $\equiv$. Clearly, $r \geq m$.

The equivalence $\equiv$ is computable, in the sense that for each equivalence class we can construct a finite automaton which accepts it. Denote by $c_1, \ldots, c_r$ the equivalence classes of $\equiv$. To each of these equivalence classes we assign a finite set of words from $\{\alpha_1, \ldots, \alpha_k\}$ as following. Given a class c_j, let $\{L_{j_1}, \ldots, L_{j_s}\}$ be the maximal set of languages from $\{L_1, \ldots, L_m\}$ such that $c_j = L_{j_1} \cap \cdots \cap L_{j_s}$. Then to c_j we assign the set $set(c_j) := \{w_{j_1}, \ldots, w_{j_s}\}$, where recall that w_j is the unique output of transducer T_j. Then, it is easy to check that

$$\tau = \bigcup_{j=1}^{s} [c_j \times set(c_j)],$$

expression which verifies the conditions of the lemma. □

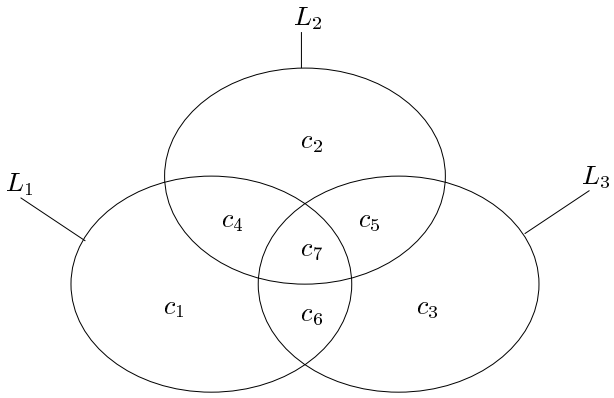

Figure 9. Example of equivalence classes for $\equiv$, when $m = 3$.

Corollary 4.2. $FC \subset Rec(X^* \times Y^*)$.

Proof:
It is a consequence of Lemma 4.1 and Mezei's characterization of recognizable sets. □

Theorem 4.1. $FC \subset IU$.

Proof:
Let τ be a rational transduction with finite codomain. By Lemma 4.1, τ can be written as

$$\tau = \bigcup_{i \in I} L_i \times R_i,$$

where I is finite, $\{L_i\}_{i \in I}$ is a family of disjoint regular languages and $\{R_i\}_{i \in I}$ is a set of finite languages. The transducer in Figure 10 is IU and realizes τ. We have denoted $I = \{1, \ldots, k\}$. The transition

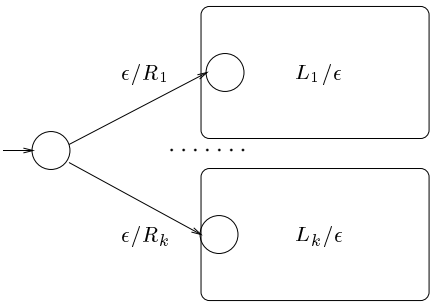

Figure 10. An IU transducer realizing a given rational transduction with finite codomain.

labelled ϵ/R_1 stands for a finite set of transitions labelled ϵ/α, for each $\alpha \in R_1$. The "block" labelled L_1/ϵ denotes a DFA which accepts L_1 and is transformed into a transducer by outputting ϵ. Since the languages $\{L_i\}_{i=1}^k$ are disjoint, input ambiguity can not occur. □

Corollary 4.3. Any FC transduction can be written as a finite union of subsequential functions.

Proof:
This is another consequence of Lemma 4.1. The transducer constructed in the proof of Theorem 4.1, and shown in Figure 10, is equivalent to a "union" of subsequential transducers. Indeed, we take each machine denoted by L_i/ϵ and we duplicate it as many times as the cardinality of R_i is. Each such machine will have output associated to every final states, all outputs equal to one word in R_i. These are the required subsequential transducers. □

Notice that obviously FC $\subset$ UA. Notice also that FC and the family of rational functions overlap, but are incomparable. We then give in Figure 11 a hierarchy describing different levels of ambiguity, in which we recall the following nomenclature:

1. `Rat`: the family of rational word relations (realized by finite transducers);
2. `FA`: the family of finitely ambiguous rational relations (the image of an input word is finite);
3. `IU`: the family of rational relations realized by input-unambiguous finite transducers (we give another machine-characterization later in this paper);
4. `UA`: the family of uniformly ambiguous rational relations (there exists a constant which upper bounds the cardinal of the image of any input word);
5. `FC`: the family of rational transductions with finite codomain;
6. `RatF`: the family of rational functions, realized by bimachines ([2, p. 123]), or equally, by unambiguous transducers;

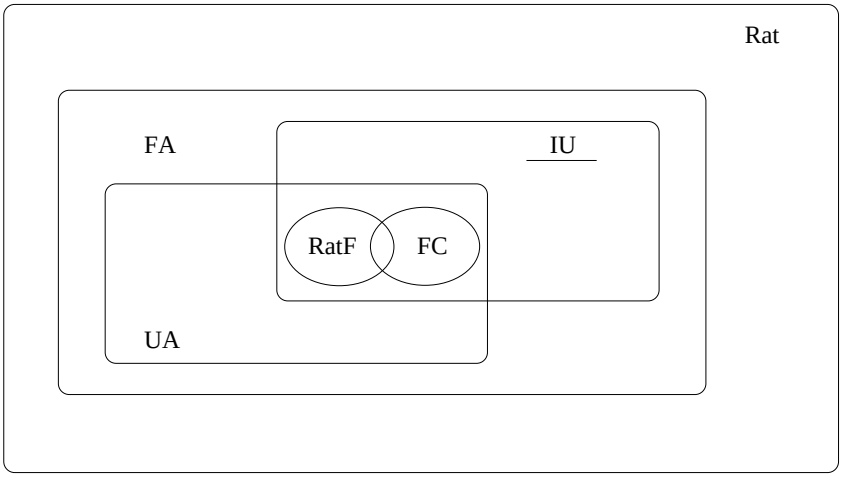

Figure 11. Different degrees of ambiguity.

5. Nondeterministic Bimachines

In the following we consider all input-unambiguous transducers to be trim and normalized according to Lemma 3.1. We are now aiming at giving a bimachine-characterization of IU.

Definition 5.1. A `bimachine` $B = (Q, P, X, Y, \delta_Q, \delta_P, q_0, p_0, \omega)$ over X and Y is composed of

- two finite sets of states Q and P,
- a finite input alphabet X and a finite output alphabet Y,
- two partial next state functions $\delta_Q : Q \times X \to Q$ and $\delta_P : X \times P \to P$,
- two initial states $q_0 \in Q$ and $p_0 \in P$,
- and a partial output function $\omega : Q \times X \times P \to Y^*$.

The next-state functions are extended to operate on words as following:

- $\forall q \in Q$ and $p \in P : \delta_Q(q, \epsilon) = q$ and $\delta_P(\epsilon, p) = p$;
- $\forall q \in Q, p \in P, a \in X$ and $w \in X^+$:

$$\delta_Q(q, wa) = \delta_Q(\delta_Q(q, w), a) \text{ and } \delta_P(aw, p) = \delta_P(a, \delta_P(w, p)).$$

Notice that function δ_P "reads" its argument word in reverse. We consider a similar extension of the output function:

- $\forall q \in Q$ and $p \in P : \omega(q, \epsilon, p) = \epsilon$;
- $\forall q \in Q, p \in P, a \in X$ and $w \in X^+$:

$$\omega(q, wa, p) = \omega(q, w, \delta_P(a, p))\omega(\delta_Q(q, w), a, p).$$

The partial word function realized by B is a function $f_B : X^* \to Y^*$, defined by $f_B(w) = \omega(q_o, w, p_0)$ if ω is defined in (q_0, w, p_0) and is undefined otherwise. Notice that always $f_B(\epsilon) = \epsilon$.

In essence, a bimachine is composed of two partial automata without final states (more precisely, all states act as final) and an output function. Indeed, (Q, X, δ_Q, q_0) will denote the `left automaton` of B and (P, X, δ_P, p_0) its `right automaton`. The bimachine B operates as illustrated in Figure 12 and explained in the following.

The symbols on the input tape are considered from left to right, starting with the leftmost one. For each considered symbol the bimachine performs a computation step yielding some output written on an output tape. In Figure 12, the current computation step considers some symbol a as the current symbol and a factorization of the input word as $w_1 a w_2$. First, both left and right automata are reset to their initial states. Then the left automaton scans w_1 from left to right, reaching an internal state q'. In the same time, the right automaton scans the subword w_2 from right to left, reaching an internal state p'. At this point, the bimachine applies the output function ω to the arguments q', a and p' and writes the result on the output tape. Next, the current position advances one position to the right and the process is repeated. The final output is the concatenation of the output for each step, as sequentially written on the output tape.

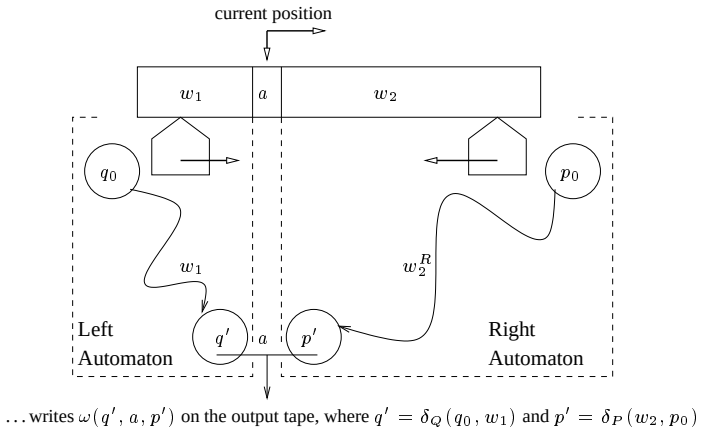

Figure 12. Computations in a bimachine.

This process is formally expressed by

$$\forall w = a_1 \dots a_n \in X^+ \text{ (where } a_i \in X, \forall i \in \{1, \dots, n\}) :$$

$$\omega(q, w, p) = \omega(q, a_1, \delta_P(a_2 \dots a_n, p))\omega(\delta_Q(q, a_1), a_2, \delta_P(a_3 \dots a_n, p)) \dots$$

$$\dots \omega(\delta_Q(q, a_1 \dots a_{n-1}), a_n, p).$$

Bimachines are of great theoretical importance since they are specifically designed to characterize the family of rational word functions, as the following result shows:

Theorem 5.1. [10, Volume A, §11.7, Theorem 7.1, p. 321] Let X, Y be finite alphabets and $f : X^* \to Y^*$ be a partial word function with $f(\epsilon) = \epsilon$. Then f is rational if and only if it is realized by some bimachine over X and Y.

However, to our knowledge, so far there have been no attempts to study nondeterministic bimachines. We distinguish 3 components of a bimachine which are candidate to nondeterminization: the left and right automata and the output function. According to this, we define the following new types of bimachines:

1. `FNObm` : with finitely nondeterministic output (at each "step" the bimachine nondeterministically writes a word on the output tape, choosing from a finite set of choices);
2. `NTbm` : with nondeterministic transitions (the two underlying automata are nondeterministic: ϵ-NFA);
3. `LNTbm` : with left nondeterministic transitions (only the "left automaton" is nondeterministic);
4. `RNTbm` : with right nondeterministic transitions (only the "right automaton" is nondeterministic);
5. `NTObm` : with both nondeterministic transitions and finitely nondeterministic output;

and we denote by FNO, NT, LNT, etc. the families of transductions realized by these types of bimachines.

On a note of caution, one must pay attention to the way in which the output functions of nondeterministic bimachines are extended. For example, following the notations at page 14, if B was a FNObm, then the extended output function (denoted here by $\overline{\omega}$) would be given by

- $\forall q \in Q$ and $p \in P : \overline{\omega}(q, \epsilon, p) = \{\epsilon\}$;
- $\forall q \in Q, a \in X$ and $p \in P : \overline{\omega}(q, a, p) = \omega(q, a, p)$;
- $\forall q \in Q, a \in X, p \in P,$ and $w \in X^+$:

$$\overline{\omega}(q, wa, p) = \overline{\omega}(q, w, \delta_P(a, p))\overline{\omega}(\delta_Q(q, w), a, p),$$

where the concatenation is viewed as a language concatenation.

If B was an NTbm, we would "extend" ω as following:

- $\forall q \in Q$ and $p \in P : \overline{\omega}(q, \epsilon, p) = \epsilon$;
- $\forall q \in Q, a \in X$ and $p \in P : \overline{\omega}(q, a, p)$ nondeterministically writes on the output tape $\omega(r, a, s)$, where $r \in \delta_Q(q, \epsilon), s \in \delta_P(\epsilon, p)$, and we follow the usual convention that $q \in \delta_Q(q, \epsilon)$ and $p \in \delta_P(\epsilon, p)$;
- $\forall q \in Q, a \in X, p \in P,$ and $w \in X^+$: $\overline{\omega}(q, wa, p)$ nondeterministically writes on the output tape $\overline{\omega}(q, w, s)\overline{\omega}(r, a, p)$, where $r \in \delta_Q(q, w)$ and $s \in \delta_P(a, p)$.

It is important to observe that at each computation step of an NTbm B, both the left and the right automata of B are "reset" to their initial state. This point is made clear in Figure 13. While reading w_1,

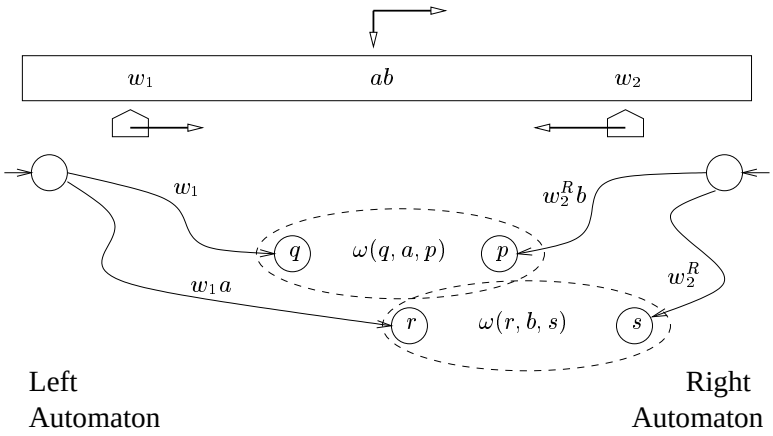

Figure 13. NTbm behavior: each computation step involves a "reset".

the left automaton reaches the state q, through the computation (path) labelled w_1. However, in the next computation step, the left automaton reads $w_1 a$ and performs a computation labelled $w_1 a$ that may not overlap with the previous computation (more precisely, the computation labelled $w_1 a$ is not prefixed by the computation labelled w_1). This is due to the fact that the left automaton is reset to the initial state before reading $w_1 a$ (it does not continue the computation from q while reading a).

A natural question which arises is what kind of transductions are realized by each of these types of nondeterministic bimachines. In the following we attempt to answer this intriguing question. We first prove that these bimachines are equally powerful.

Theorem 5.2. $FNO = NT = LNT = RNT = NTO$.

In other words, it does not matter which component of the bimachine is nondeterministic. For this reason, we are allowed to employ the term `nondeterministic bimachine` in a generic sense.

Proof:
We prove the equality $FNO = LNT$, the others being left to the reader.

We first prove that $FNO \subseteq LNT$. Let $B = (Q, P, X, Y, \delta_Q, \delta_P, q_0, p_0, \omega)$ be a $FNObm$, i.e., a bimachine as in Definition 5.1, with the exception that the output function ω is defined as

$$\omega : Q \times X \times P \to \mathcal{FP}(Y^*),$$

where by $\mathcal{FP}(Y^*)$ we understand the set of all finite languages over Y (the set of `finite parts` of Y^*). When this bimachine scans some input, for each computed triplet $(q, a, p) \in Q \times X \times P$ it nondeterministically chooses a word in $\omega(q, a, p)$ and writes it on the output tape. In other words, at each "writing", the bimachine has a finite number of choices for the word to be written on the output tape. We define an equivalent bimachine $(Q', P, X, Y, \delta_{Q'}, \delta_P, q'_0, p_0, \omega')$ in $LNTbm$ as following.

1. We first set $Q' := Q$ and $\delta_{Q'} = \delta_Q$.
   In the following, `by replacing a state` $q \in Q'$ `with the set of new states` $\{q_1, \ldots, q_k\}$ *we understand to add for each input transition into q identical input transitions into all* $q_1, \ldots q_k$*, for each output transition from* q *identical output transitions from all* $q_1, \ldots, q_k$ *and finally to remove* q *and all its adjacent transitions. (if* q *had loop transitions, we would process them differently: we would attach these loops to each of* $q_1, \ldots, q_k$*).*

2. For every state $q \in Q$ perform the following:

   Set $K_q = \emptyset$. For each tuple $(q, a, p) \in Q \times X \times P$ with $\mid \omega(q, a, p) \mid > 1$, denote $\omega(q, a, p) = \{v_1^p, \ldots, v_k^p\}$ and add to K_q a set of new states: $K_q := K_q \cup \{q_1^p, \ldots, q_k^p\}$. Then replace (in the sense previously described) q by the set of new states K_q and set $\omega'(q_i^p, a, p) = v_i^p$ for all $i \in \{1, \ldots, k\}$, and this for all states p which have been processed.

3. If in the previous step the start state q_0 has not been replaced, then set $q'_0 = q_0$. Otherwise, if q_0 has been replaced by a set K_0 of new states then add a new start state q'_0 and set $\delta_{Q'}(q'_0, \epsilon) = K_0$.

Notice that the right automaton has not been changed and the left automaton has become an ϵ-NFA, with ϵ-transitions present only among the output transitions of q'_0, eventually. Notice also that ω' has become a partial function mapping $Q \times X \times P$ into Y^*.

Let an input word uav be with $u, v \in X^*$ and $a \in X$. One can carefully check that we have in B the computations $\delta_Q(q_0, u) = q$, $\delta_P(v, p_0) = p$ and $\omega(q, a, p) = \{w_1, \ldots, w_k\}$ `if and only if` we have in B' the computations $\delta_{Q'}(q'_0, u) \supseteq \{q_1^p, \ldots, q_k^p\}$ and $\omega'(q_i^p, a, p) = w_i$ for all $i \in \{1, \ldots, k\}$. This proves the equivalence of the two bimachines.

We now prove that $FNO \supseteq LNT$. Let $B = (Q, P, X, Y, \delta_Q, \delta_P, q_0, p_0, \omega)$ be a LNT bimachine, i.e., a bimachine as in Definition 5.1, with the exception that the left automaton is an ϵ-NFA. The operation of such bimachine is obvious: everything works as in the standard bimachine, with the exception that at each step we must consider more than one configuration (i.e., argument of the output function), since the computations of the left automaton are nondeterministic. The idea for constructing an equivalent $FNObm$

is to proceed by the determinization of the left automaton, moving the nondeterminism to the output function. We construct an equivalent FNO bimachine $B' = (Q' \subseteq \mathcal{P}(Q), P, X, Y, \delta_{Q'}, \delta_p, \{q_0\}, p_0, \omega')$ as follows.

We consider the extended function $\delta_Q^* : \mathcal{P}(Q) \times X^* \to \mathcal{P}(Q)$ in the standard way and define

1. $Q' := \{\delta_Q^*(q_0, w) \mid w \in X^*\}$, i.e., Q' is a set of subsets of states of Q, namely of those subsets each containing all the states which are reached nondeterministically, starting from the initial state and reading some input;
2. $\forall K \in Q', \forall a \in X : \quad \delta_{Q'}(K, a) = \bigcup_{q \in K} \delta_Q^*(q, a)$ (notice that δ_Q^* takes into consideration ϵ transitions as well);
3. $\forall K \in Q', a \in X, p \in P : \quad \omega'(K, a, p) = \bigcup_{q \in K} \omega(q, a, p)$ – which is obviously finite.

Then the left automaton of B' is deterministic, the output function ω' is nondeterministic with a finite number of choices, hence B' is a FNO bimachine.

Let $uav \in X^*$ be an input with $u, v \in X^*$ and $a \in X$. In the bimachine B we have $\delta_Q^*(q_0, u) = \{q_1, \ldots, q_k\}$, $\delta(v, p_0) = p$ and $\omega(q_i, a, p) = v_i$, for $i \in \{1, \ldots, k\}$ `if and only if` in the bimachine B' we have $\delta_{Q'}(\{q_0\}, u) = V$ and $\omega'(V, a, p) = \{v_1, \ldots, v_k\}$. This proves the bimachine equivalence. □

It has been shown in [24] that the scanning direction of the reading heads of a deterministic bimachine does not count. It is natural to question whether this property still holds for nondeterministic bimachines.

Theorem 5.3. The parsing direction of the reading heads of a nondeterministic bimachine does not matter.

I.e., convergent, left sequential, right sequential, and divergent nondeterministic bimachines have equal power.

Proof:
The idea of the proof is to use FNO bimachines and adapt the proof in [24, Theorem 16, p. 135] to the nondeterministic case. The details are left to the reader. □

We are now ready to state one of the main results of this paper, namely a bimachine characterization of input-unambiguous rational transductions.

Theorem 5.4. A transduction τ with $\tau(\epsilon) = \epsilon$ is IU rational `if and only if` it is realized by a nondeterministic bimachine.

Proof:
We first prove the implication to the right (only if). For that we follow an idea similar to, but nevertheless different from, the proof that deterministic bimachines characterize rational functions – as found in [2, Theorem 5.1, p.125].

Let τ be realized by $T = (Q, X, Y, E, q_0, \{q_0, q_f\})$: a trim, IU, finite transducer normalized according to Lemma 2. Notice that this transducer does not have ϵ-input transitions, since the input labels of its transitions are letters of the alphabet X.

By a computation in T, denoted by $*(q, u, \alpha, p)$, we understand a computation which starts in q and ends in p while reading u from the input tape and writing α on the output tape. Based on relation (1) at page 3, which holds for every ϵ-free UNFA, we infer that for all $u, v \in X^*$, if there exist two computations in T:

$$*(q_0, u, \alpha, p), \quad *(p, v, \beta, q_f),$$

then p is uniquely determined by u and v. Define the following subsets of Q:

- $U_u := \{p \mid \exists \alpha \in Y^* : \ *(q_0, u, \alpha, p)$ is a valid computation in $T\}$,
- $V_v := \{p \mid \exists \beta \in Y^* : \ *(p, v, \beta, q_f)$ is a valid computation in $T\}$,
- $\mathcal{U} := \{U_u\}_{u \in X^*}, \ \mathcal{V} := \{V_v\}_{v \in X^*}$ (clearly, both are finite).

The FNO bimachine equivalent to T is $B = (\mathcal{U}, \mathcal{V}, X, Y, \delta_{\mathcal{U}}, \delta_{\mathcal{V}}, \{q_0\}, \{q_f\}, \omega)$, where the left and right automata and the output function are defined as following:

1. $(\mathcal{U}, X, \delta_{\mathcal{U}}, \{q_0\})$ is the `left automaton`, where
$$\delta_{\mathcal{U}}(U, a) := \bigcup_{q \in U} \{p \mid (q, a, p) \in E\},$$
2. $(\mathcal{V}, X, \delta_{\mathcal{V}}, \{q_f\})$ is the `right automaton`, where
$$\delta_{\mathcal{V}}(a, V) := \bigcup_{q \in V} \{p \mid (p, a, q) \in E\},$$
3. $\omega : \mathcal{U} \times X \times \mathcal{V} \to \mathcal{PF}(Y^*)$ is the `output function`, given by
$$\omega(U, a, V) := \{\alpha \mid (\delta_{\mathcal{V}}(a, V) \cap U, a, \alpha, V \cap \delta_{\mathcal{U}}(U, a)) \in E\} \ .$$
In the relation defining ω, we have used that $|V \cap \delta_{\mathcal{U}}(U, a)| \leq 1$ and $|\delta_{\mathcal{V}}(a, V) \cap U| \leq 1$ (from a previous observation), and we followed the convention of identifying a singleton set with its element.

The operation of B is depicted in Figure 14, where the left automaton reads u, the right automaton reads v and ω nondeterministically writes one of $\{\alpha_1, \ldots, \alpha_k\}$ on the output tape (the input word is uav^R).

For proving the other implication, consider a FNO bimachine $B = (Q, P, X, Y, \delta_Q, \delta_P, q_0, p_0, \omega)$. We construct the finite transducer $T = (Q \times P \cup \{q_0'\}, X, Y, E, q_0', F)$, given by

- q_0' is a new state, the initial state of T;
- $F := \{(q, p_0) \mid q \in Q\}$ is the set of final states;
- the finite set of transitions E is given by:

  $\forall p \in P : \ \big(q_0', \epsilon, \epsilon, (q_0, p)\big) \in E$, and
  $\forall a \in X$, if $\delta_Q(q, a) = q'$ and $\delta_P(a, p) = p'$, then
  $$\forall \alpha \in \omega(q, a, p) : \big((q, p'), a, \alpha, (q', p)\big) \in E.$$
  Notice that E is finite for the reason that $|\omega(q, a, p)| < \aleph_0$, which holds by the fact that B is a FNO bimachine.

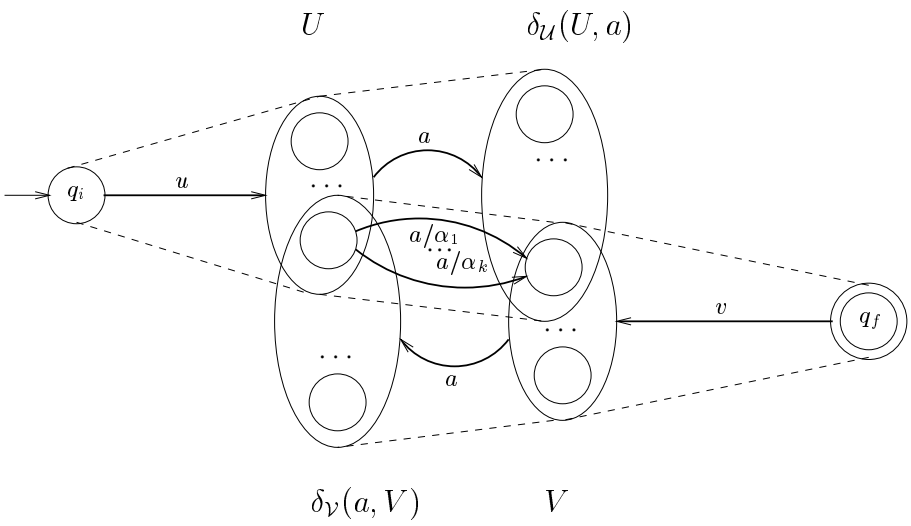

Figure 14. Bimachine construction.

It can easily be checked that T realizes the same transduction as B. It remains to prove that T is IU. Let $u = a_1 \dots a_k$ be a word in the domain of the transduction realized by T, with $a_1, \dots, a_k \in X$. In B, assume that $\delta_Q(q_0, a_1) = q_1, \dots, \delta_Q(q_{k-1}, a_k) = q_k$ and that $\delta_P(a_k, p_0) = p_1, \dots, \delta_P(a_1, p_{k-1}) = p_k$. Then, in the input automaton of T there exist the following successful computation:

$$q_0' a_1 \dots a_k \vdash (q_0, p_k) a_1 \dots a_k \vdash \dots \vdash (q_{k-1}, p_1) a_k \vdash (q_k, p_0).$$

Moreover, due to the determinism of δ_Q and δ_P and also to the construction of T, this is the unique successful computation of the input automaton of T when it reads u. Notice that we can have, for example, $\delta_Q(q_0, a_1) = q_1$, $\delta_P(p_{k-1}, a_1) = p_k$ and, say, $\delta_P(r, a_1) = p_k$ for some $r \in P, r \neq p_{k-1}$. That would imply input-nondeterministic transitions in T: $(q_0, p_k)a_1 \vdash (q_1, p_{k-1})$ and $(q_0, p_k)a_1 \vdash (q_1, r)$. However, only p_{k-1} can be reached from p_0 by the input $a_k \dots a_2$. This proves the input unambiguity of T. □

By a finite substitution from X to Y we understand a function $\sigma : X \to \mathcal{FP}(Y^*)$. We extend σ to X^* in a natural way (hence becoming a monoid morphism).

Corollary 5.1. A transduction $\tau : X^* \to Y^*$ is IU if and only if there exists a rational function $\mu : X^* \to Z^*$ and a finite substitution $\sigma : Z^* \to \mathcal{FP}(Y^*)$ such that $\tau = \sigma \circ \mu$.

Proof:
For the implication to the right, we use the representation of IU transductions by FNObm's. If B is a FNO bimachine with the output function $\omega : Q \times X \times P \to \mathcal{FP}(Y^*)$, we transform ω into $\omega' : Q \times X \times P \to (Q \times X \times P)^*$, given by

$$\omega'(q, a, p) := (q, a, p),$$

and we define the substitution $\sigma : (Q \times X \times P) \to \mathcal{FP}(Y^*)$ to be given by $\sigma((q,a,p)) := \omega(q,a,p)$. Obviously, $Z = Q \times X \times P$ and since a classical bimachine realizes a rational function, the conclusion follows.

The other implication is proven in the reversed way, starting with a bimachine and a finite substitution and construct a FNObm equivalent to their composition. The details are left to the reader. □

Remark 5.1. It is decidable whether a nondeterministic bimachine is single-valued (realizes a rational function).

Proof:
Since functionality is decidable for finite transducers ([27]), in order to decide whether a nondeterministic bimachine is functional or not it suffices to construct an equivalent IU transducer as in the proof of Theorem 5.4 and decide its functionality.

Alternatively, we may use the fact that any nondeterministic bimachine can be transformed into a FNObm, and for such bimachine it suffices to analyze its output function. □

We next give a necessary condition for a rational transduction to be IU.

Corollary 5.2. A rational transduction $\tau : X^* \to Y^*$ is IU only if there exist two constants M and N such that

1. $\forall u \in X^* : \ |\tau(u)| \leq M \cdot \sharp(u)$;
2. $\forall u \in X^*, \ \forall v \in \tau(u) : \ \sharp(v) \leq N \cdot \sharp(u)$.

In other words, the number of outputs for a given input of an IU transduction is a linear function of the length of the input and the length of any output is also a linear function of the length of the input.

Proof:
Consider τ being realized by a FNObm, and denote by N the length of the longest output of the bimachine in a single step, and by M the maximum number of output choices of the bimachine in a single step. The conclusion follows from the fact that the number of computation steps of a bimachine is exactly the length of the input word (in a successful computation). □

The reciprocal of the previous corollary does not hold, as the transduction (2) at page 9 shows: the transduction is not IU, however it verifies the conditions 1 and 2 of Corollary 5.2.

Remark 5.2. Another surprising consequence of Corollary 5.1 and Theorem 4.1 is that any FC transduction can also be written as a composition of a rational function and a finite substitution.

Definition 5.2. An `input-deterministic` (ID) transducer is a tuple $T = (Q, X, Y, \delta, \omega)$ where X, Y are alphabets, Q is a finite set of states, and

$$\delta : Q \times X \to Q, \text{ and } \omega : Q \times X \to \mathcal{FP}(Y^*)$$

are partial functions with the same domain, denoting the transition and the output function.

In other words, an ID transducer is similar to a sequential transducer, with the exception that reading an input letter leads to a finite number of output choices. Notice that a transducer is ID if and only if its input automaton is deterministic – hence justifying its name. As usual, we define the family of ID transductions to be the family of all transductions that are realized by ID transducers.

Lemma 5.1. A transduction is ID if and only if it is the composition of a sequential transduction and a finite substitution.

Proof:
The proof is similar to the proof of Corollary 5.1. The details are left to the reader. □

Remark 5.3. Although both FC and ID are included in IU, there is no relation of inclusion between FC and ID. For example, the transduction $\mu : \{a\}^* \to \{a\}^*$ given by

$$\forall n \geq 1 : \ \ \mu(a^n) = \bigcup_{i=1}^{n} \{a^i\}$$

is in ID but not in FC; whereas the transduction $\nu : \{a\}^* \to \{a, b\}^*$ given by

$$\forall n \geq 1 : \ \ \nu(a^n) = \begin{cases} a, & \text{if } n \text{ is even,} \\ b, & \text{otherwise,} \end{cases}$$

is in FC but not in ID. Consequently, we infer that both FC and ID are strictly included in IU.

Theorem 5.5. Let $\tau : X^* \to Y^*$ be a transduction with $\tau(\epsilon) = \epsilon$. Then τ is an IU transduction if and only if there are a right sequential function $\mu : X^* \to Z^*$ and an ID transduction $\nu : Z^* \to Y^*$ such that $\tau = \nu \circ \mu$. Moreover, μ can be chosen to be total and length preserving.

Intuitively, the sequential transducer represents the set of unique successful paths of the unambiguous transducer, whereas the ID transducer represents the nondeterminism of the output process.

Proof:
We prove the implication to the right (only if). According to Theorem 5.4, since τ is in IU, there exists a FNObm B which realizes it. Let $A = (Q, X, \delta_Q, q_0)$ be the left automaton and $B = (P, X, \delta_P, p_0)$ be the right automaton of B and ω be its output function. We denote $Z := X \times P$, and we transform B into a right sequential transducer, by adding to it an output function ω', given by

$$\omega' : P \times X \to Z^*, \ \ \omega'(p, a) := (a, p).$$

in other words, we modify B to "dump" its computation. We set $\mu := |B|$ where B is now a right sequential transducer.

Next we perform two modifications to the right automaton A. First, we change A to read symbols from Z, and advance its computations according to the "letter component" of the symbols in Z. For example, if initially we had $\delta_Q(q_1, a) = q_2$, we now have $\delta_Q(q_1, (a, p)) = q_2$, for all $p \in P$. Secondly, we add an output function ω'' to A, given by

$$\omega'' : Q \times Z \to Y^*, \ \ \omega''(q, (a, p)) := \omega(q, a, p) \ \ ,$$

where recall that ω was the output function of B. then, A becomes an ID transducer and we set $\nu := |A|$. It is easy to see that $\tau = \nu \circ \mu$ and that μ is length preserving. Certainly, μ can be made total, by making the DFA B total in the first place. We depicted the construction in Figure 15, where by SEQ we denote the right sequential transducer B and by ID we denote the input-deterministic transducer A.

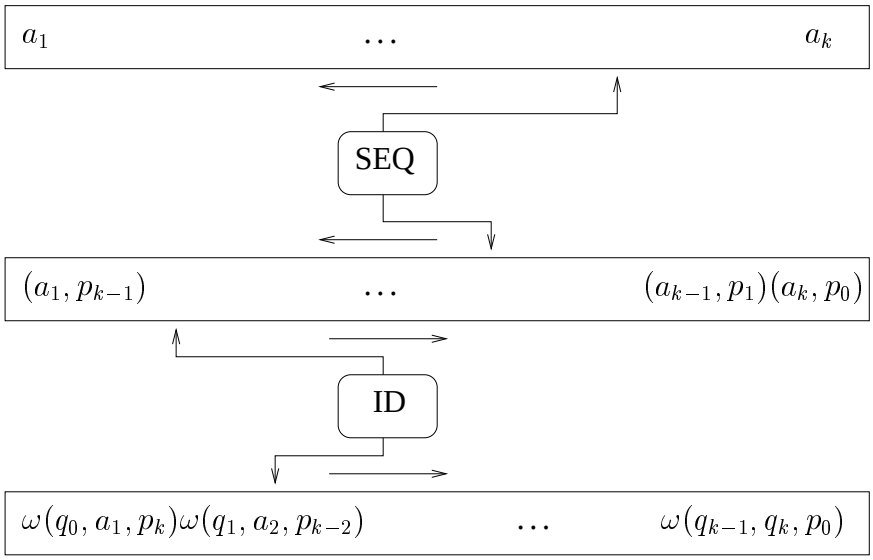

Figure 15. A characterization of IU transductions.

Conversely, we recall from Corollary 5.1 that an ID transduction is the composition of a left sequential transduction and a finite substitution. We also know that the composition of a right sequential transducer with a left sequential transducer is a rational function. Therefore, we obtain the composition of a rational function and a finite substitution, which is exactly an IU transduction, as proven in Corollary 5.1. □

It is also worth mentioning that a transduction is IU if and only if it is the composition of a left sequential transduction and a right ID transducer. Here, by a right ID transducer we understand a transducer that scans the input from right to left and writes the output from right to left as well. It is apparent by now the similarity between this characterization and the characterization of rational functions by right and left sequential transductions.

6. Restricted Nondeterministic Bimachines

In the previous section we have introduced nondeterministic bimachines with a special behavior: at each computation step, these bimachines perform a "reset", i.e., they set their underlying automata to be in initial state. Then a natural question occurs, that is, what would happen if we inhibit the "reset"? This leads to the definition of another type of nondetrministic bimachine: a `restricted nondeterministic bimachine`. At each step, these bimachines are forced to continue their computation from the states reached at the previous step.

Definition 6.1. A restricted nondeterministic bimachine (RNTbm) is a bimachine with nondeterministic transitions (NTbm) and multiple initial states $B = (Q, P, X, Y, \delta_Q, \delta_P, I_Q, I_P, \omega)$, where the output function is extended as following:

- $\forall q \in Q, p \in P: \ \omega(q, \epsilon, p) = \{\epsilon\}$;
- $\forall w = a_1 \dots a_n \in X^+$ (where $\forall i \in \{1, \dots, n\} : a_i \in X$),
  $\forall q_0 \in I_Q, p_0 \in I_P$, $\omega(q_0, w, p_0)$ is given by:

  $\{\omega(q_0, a_1, p_{n-1})\omega(q_1, a_2, p_{n-2}) \dots \omega(q_{n-2}, a_{n-1}, p_1)\omega(q_{n-1}, a_n, p_0) \mid$
  $q_1 \in \delta_Q^*(q_0, a_1), \dots, q_{n-1} \in \delta_Q^*(q_{n-2}, a_{n-1}),$
  $p_1 \in \delta_P^*(a_n, p_0), \dots, p_{n-1} \in \delta_P^*(a_2, p_{n-2})\}$.

Notice that by this behavior, the bimachine still operates nondeterministically. However, the current states of its automata depend on the previous current states. We will see that, although this seems like a restriction, RNTbm's have a greater power than NTbm's. Notice also that we allow multiple initial states – for improving the formalism. At the beginning of the operation, a RNT bimachine sets itself nondeterministically into two initial states corresponding to its left and right automata.

Theorem 6.1. A transduction τ with $\tau(\epsilon) = \epsilon$ is in FA if and only if it is realized by a RNTbm.

Proof:
We give a sketch of the proof. Suppose $\tau \in FA$. Then τ is realized by a transducer $T = (Q, X, Y, E, q_0, F)$ with no ϵ-input loops. Moreover, we can assume that T is normalized according to Lemma 3.1 since the construction of the lemma works for any transducer with no ϵ-input loops. This implies among others that $E \in Q \times X \times Y^* \times Q$ and $F = \{q_0, q_f\}$. In an initial stage, we modify T to obey the following property:

$$(p, a, \alpha, q), (p, a, \beta, q') \in E \Rightarrow \alpha = \beta, \tag{3}$$

where $a \in X$, $\alpha, \beta \in Y^*$, $p, q, q' \in Q$. Suppose this property does not hold for p, and we have a situation as depicted in Figure 16 (A). In this situation we perform the construction in Figure 16 (B). We duplicate states and separate transitions in such way that any input label is associated to a same output label across all outward transitions of every state. The construction holds for the case of the initial state as well: if the initial state q_0 is duplicated, then we obtain a set of initial states, I_Q.

Notice that the obtained transducer is still normalized, except possibly for multiple initial states. Then, the RNT bimachine realizing τ is $B = (Q, Q, X, Y, \delta_Q, \delta_P, I_Q, q_f, \omega)$, given by:

$$\forall x \in X \cup \{\epsilon\}, q \in Q : \delta_Q(q, x) = \{q' \mid \exists \alpha \in Y^* : (q, x, \alpha, q') \in E\};$$

$$\forall x \in X \cup \{\epsilon\}, p \in Q : \delta_P(x, p) = \{p' \mid \exists \alpha \in Y^* : (p', x, \alpha, p) \in E\};$$

$$\forall q, p \in Q, a \in X: \ \omega(q, a, p) = \alpha \text{ if } (q, a, \alpha, p) \in E.$$

One can show that B is equivalent with T, that is, it realizes τ.

For the other implication, given a RNTbm $B = (Q, P, X, Y, \delta_Q, \delta_P, I_Q, I_P, \omega)$, it is enough to construct an equivalent transducer. Indeed, if such transducer exists, it certainly realizes a FA transduction, due to the fact that the image of any input through the bimachine B is finite: the number of possible

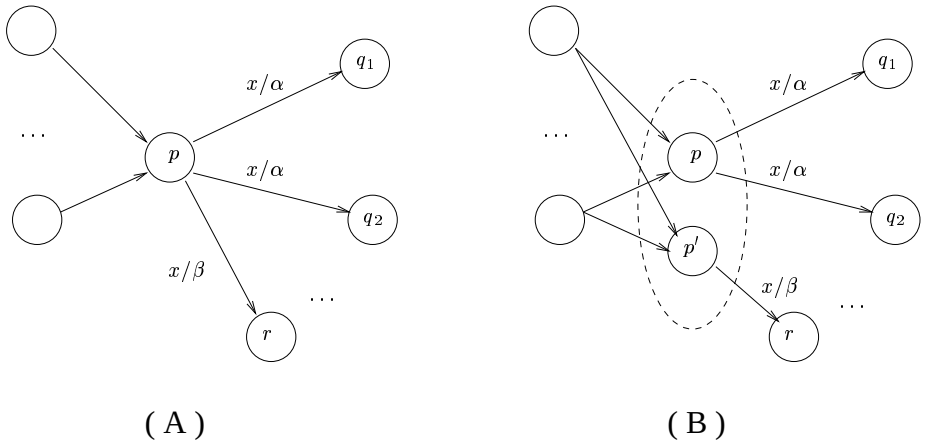

Figure 16. Addition of extra states to ensure the property.

output words is bound by a linear function on the length of the input word. Indeed, by Definition 6.1, the bimachine writes on the output tape only after it consumes an input symbols. Then, we just need to prove that B realizes a rational transduction. Consider the transducer $T = (Q \times P \cup \{q_0'\}, X, Y, E, q_0', F)$ where

- q_0' is a new state having ϵ/ϵ transitions to each state (q_0, p), with $q_0 \in I_Q$ and $p \in P$;
- $F = \{(q, p_0) \mid q \in Q, p_0 \in I_P\}$;
- $E = \{((q,p), a, \alpha, (q',p')) \mid \omega(q,a,p') = \alpha$ and $q' \in \delta_Q^*(q,a), p \in \delta_P^*(a,p')\}$, where, as usual, by δ_Q^* we understand the function δ_Q extended to words.

We next ensure that T accepts (ϵ, ϵ) by setting q_0' to be final as well. Notice that the obtained transducer has no ϵ-input transitions, except for those emerging from the initial state (which was expected). It can be shown that the transducer T is equivalent to B. □

In the following we take a transduction in FA \ IU and construct a RNT bimachine that realizes it. By Theorem 5.4, this transduction can not be realized by an NT bimachine.

Example 6.1. Consider the transduction defined on page 9

$$\tau = \{(a^n, x^n) \mid n \geq 0\} \cup \{(a^n, y^n) \mid n \geq 1\} \tag{4}$$

which is in FA but not in IU. This transduction is realized by the normalized transducer in Figure 17 left. On the right is shown an equivalent transducer (with more than one initial states) that has the property (3) at page 24.

In Figure 19 we depicted the left and right automata (as given by the construction of Theorem 6.1) of a corresponding RNT bimachine whose output function is given by the table in Figure 18.

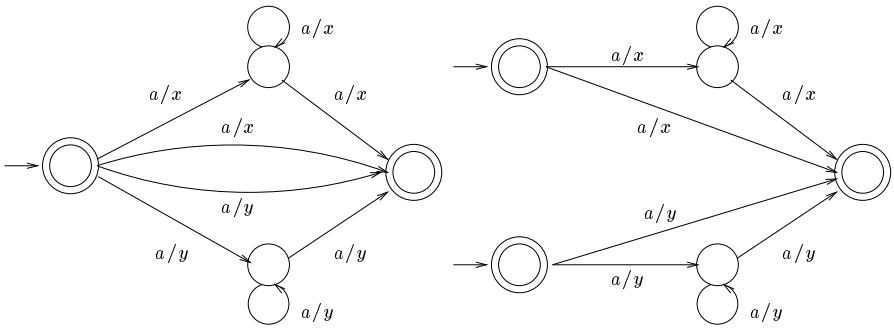

Figure 17. A normalized transducer realizing τ.

ω	p_0	p_1	p_2	p_3	p_4
q_0^1	x	x			
q_0^2	y		y		
q_1	x	x			
q_2	y		y		
q_f					

Figure 18. The output function for Example 6.1.

7. Conclusion and Further Work

The goal of this paper has been twofold: to introduce nondeterministic bimachines and to study rational relations with finite codomain. Nondeterministic bimachines can realize FC relations; however, they can do better than that: they exactly represent the family of transductions that are the composition of rational functions and finite substitutions. The transducer counterpart of these machines is the input-unambiguous transducer, which is a slight variation of the classical notion of unambiguous transducer. FC relations are recognizable and they have a particular "Mezei representation", as a finite union of blocks with certain properties: their left components are disjoint and their right ones are finite. This leads in a natural way to the representation of FC relations as a finite union of subsequential functions – notice the parallel with the uniformly ambiguous rational relations, that are finite unions of rational functions.

Nondeterministic bimachines can work in two "modes": with or without reset. We have proven that suppressing the reset in between computation steps increases their power: they now characterize the family of finitely ambiguous transductions.

Along with these lines, further questions can be explored. For example, we still do not have a decidability result about classifying a rational transduction as being IU or not. Or, since we have proven that any FC transduction can also be represented as a composition of a rational function and a finite

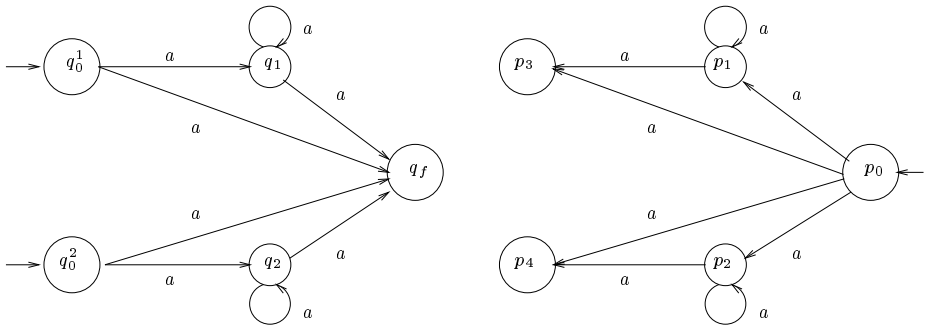

Figure 19. The transition graph of a bimachine realizing τ.

substitution, it is natural to question whether this representation can efficiently be given. Moreover, a few important matters which have been addressed here may be studied in more detail. For example, we believe that the representation of FC transductions as a finite union of subsequential functions requires further attention. More specifically, since it is a representation and not a characterization, we may study these finite unions in their own rights.

Finally, we believe that all major rational families of transductions have a "bimachine" counterpart. In particular, we left for immediate work the study of "ϵ-RNT" bimachines (i.e., RNT bimachines with ϵ-advancement) that we believe to characterize the entire family of rational relations. A bimachine-induced hierarchy of rational relations is our next goal.

References

[1] M.-P. Beal, O. Carton, C. Prieur, J. Sakarovitch: Squaring transducers: an efficient procedure for deciding functionality and sequentiality. LNCS 1776, Springer, Berlin, 2000, 397–406.

[2] J. Berstel: *Transductions and Context-Free Languages*. B.G. Teubner, Stuttgart, 1979.

[3] J. Berstel, C. Reutenauer: *Rational Series and their Languages*. Springer-Verlag, Berlin, 1988.

[4] D. Breslauer: The suffix tree of a tree and minimizing sequential transducers. *Theoretical Computer Science*, 191 (1998), 131–144.

[5] C. Choffrut: Une characterisation des fonctions sequentielles et des fonctions sous-sequentielles en tant que relations rationelles. *Theoretical Computer Science*, 5 (1977), 325–337.

[6] C. Choffrut: *Contribution a l'etude de quelques familles remarquables de fonctions rationnelles*. These d'Etat, Universite Paris VII, 1978.

[7] C. Choffrut: A generalization of Ginsburg and Rose's characterization of GSM mappings. LNCS 71, Springer-Verlag, Berlin, 1979, 88–103.

[8] C. Choffrut: Minimizing subsequential transducers: a survey. *Theoretical Computer Science*, 292 (2003), 131–143.

[9] C. Choffrut, K. Culik: Properties of finite and pushdown transducers. *SIAM Journal on Computing*, 12, 2 (1983).

[10] S. Eilenberg: *Automata, Languages and Machines*, vol. A. Academic Press, New York and London, 1974.

[11] C.C.Elgot, J.E. Mezei: On relations defined by generalized finite automata. *IBM Journal of Research and Development*, 9 (1965), 47–65.

[12] J. Hopcroft, J. Ullman: *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Massachusetts and London, 1979.

[13] S. Konstantinidis: Transducers and the properties of error-detection, error-correction, and finite-delay decodability. *Journal of Universal Computer Science*, 8 (2002), 278–291.

[14] A. Mateescu, A. Salomaa: Aspects of classical language theory. In [22], vol. 1, 175–251.

[15] J. McKnight: Kleene quotient theorems. *Pacific Journal of Mathematics*, 14 (1964), 1343–1352.

[16] M. Mohri: Minimization of sequential transducers. LNCS 807, Springer-Verlag, Berlin, 1994, 151–163.

[17] M. Mohri: Finite-state transducers in language and speech processing. *Computational Linguistics*, 23 (1997), 269–311.

[18] M. Mohri: Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14, 6 (2003), 957–982.

[19] M. Nivat: Transductions des langages de Chomsky. *Ann. de l'Inst. Fourier*, 18 (1968), 339–456.

[20] C. Reutenauer: Subsequential functions: characterizations, minimization, examples. LNCS 464, Springer-Verlag, Berlin, 1990, 62–79.

[21] C. Reutenauer, M.-P. Schutzenberger: Minimization of rational word functions. *SIAM Journal on Computing*, 30, 4 (1991), 669–685.

[22] G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.

[23] J. Sakarovitch: *Éléments de Théorie des Automates*, Vuibert Informatique, Paris, 2003.

[24] N. Santean: Bimachines and structurally-reversed automata. *Journal of Automata, Languages and Combinatorics*, 9, 1 (2004), 121–146.

[25] M.-P. Schutzenberger: On the definition of a family of automata. *Information and Control*, 4 (1961), 245–270.

[26] M.-P. Schutzenberger: A remark on finite transducers. *Information and Control*, 4 (1961), 185–196.

[27] M.-P. Schutzenberger: Sur les relations rationelles. LNCS 33, Springer-Verlag, Berlin, 1975, 209–213.

[28] M.-P. Schutzenberger: Sur une variante des fonctions sequentielles. *Theoretical Computer Science*, 4 (1977), 47–57.

[29] S. Yu: Regular Languages. In [22], vol. 1, 209–213.